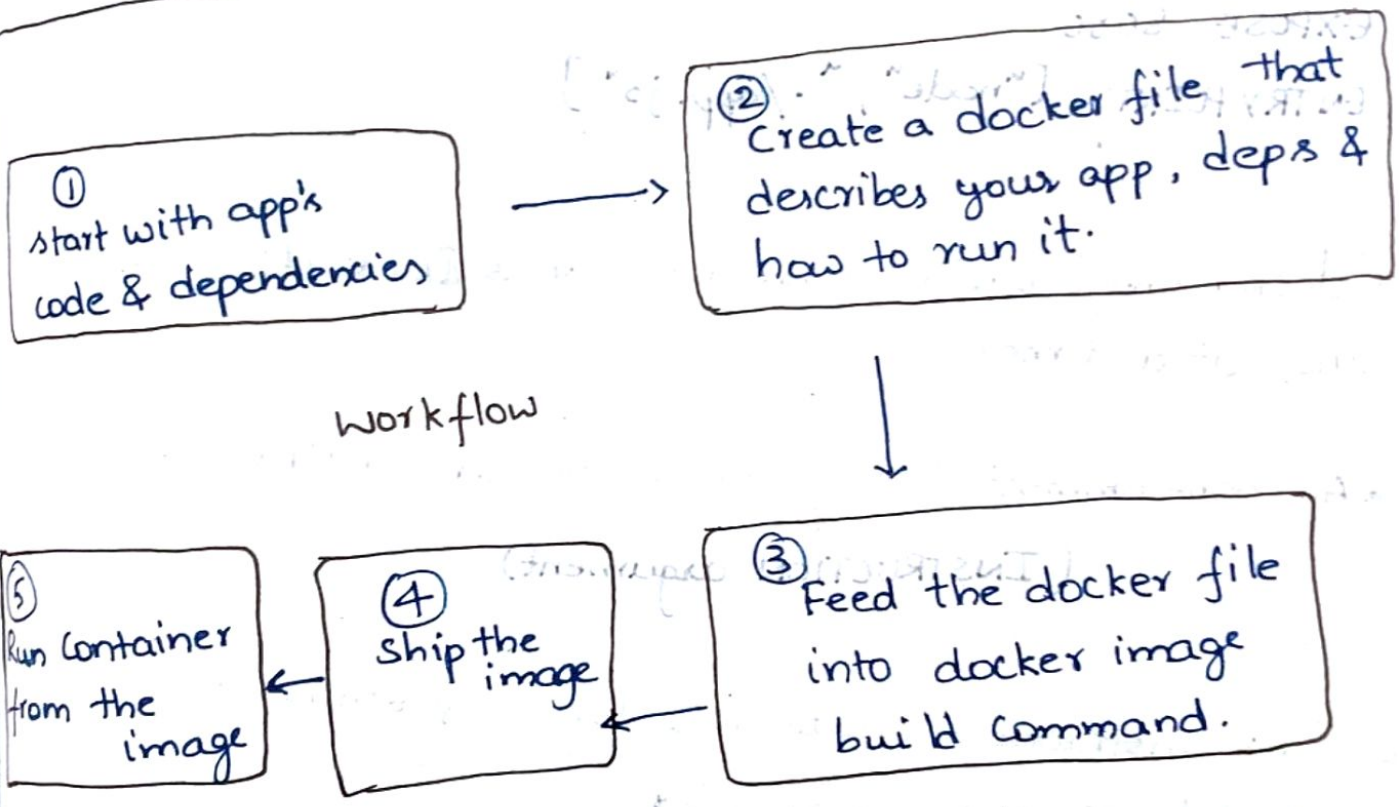
Containerizing an App:-

Docker is all about

taking applications & running them in Containers

all about making apps simple to build, ship & run.



Dockerfile:- starting point for creating a container image.

build context :- Directory containing application & dependencies.
common practice to keep Dockerfile in root.

Example Dockerfile:-

```
FROM alpine
LABEL maintainer = "Suhas gumma"
RUN apk add--update nodejs nodejs-npm
COPY ./src
WORKDIR /src
RUN npm install
EXPOSE 8080
ENTRY POINT ["node", "./app.js"]
```

---

* "Docker image build" command parses Dockerfile one line at a time.

* All non-comment lines of Dockerfile are instructions. (INSTRUCTION argument)

* Some instructions create new layers & others just add meta data to image config files.

* View instructions using "docker image history"

Multi-Stage Builds

Best Practices :-

Leverage the build cache :-

* Docker stores the layers in cache & check at the time of build, if the layer is already built.
* But, if the first layer is not in cache, it will not check for next layers
* That is why you build the first layer which is more likely to be found in cache. (If possible)

Squash the image into single layer when it is {ideal}

FROM instruction → Base image for the new Image

RUN instruction → Run commands inside the image, creates a new layer.

COPY instruction → COPY creates a new ~~file~~ layer.

EXPOSE → documents the network port that application uses

ENTRYPOINT → default application to run when the image started as a container.